Group #1

Absalon, Arc Nil

Aguirre, Julian Anthony

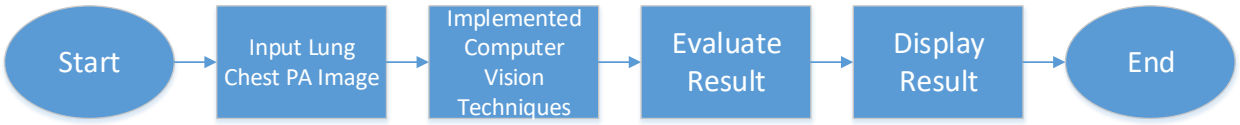
Dela Cruz, Exekiel

Paborada, Cember Mae

Paniergo, Hector

Overview

This documentation provides analyzation of the problem, the implemented solution how to solve the problem, and how the solution is implemented in Python. The main goal of this program is to process a DICOM image and detect its edges using computer vision techniques.



Activity Overview

Problem Analysis

In the medical field, the Doctor must accurately diagnose a patient's condition, especially for organs, such as Lungs. There are many methods to diagnose a condition, but some might need surgical procedures to get some samples. One way to solve this problem is to create something that can accurately generate processed images so that doctors can make accurate diagnoses. In this way, it is less dangerous than any procedure.

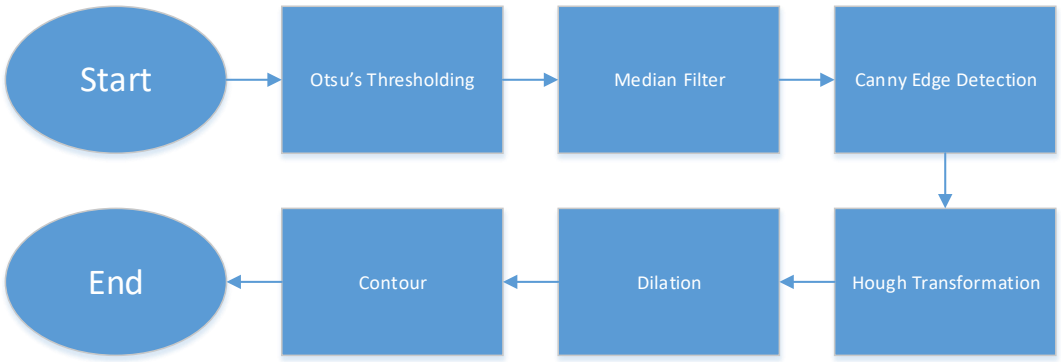
However, images do not come in the way we want, such as having too much noise or the part of the body where interest has poor visibility. We need to highlight the ROI in an image. Fortunately, there are data images available online, that we can use. We chose the "Lung X-ray Chest PA only" Image from TCIA. The image comes in DICOM format.

Solution

There are many medical imaging techniques available today and one of those is Edge Detection. Looking at the study of Jadwaa (2022), she can perform edge detection by Canny Edge to highlight the Region of Interest (Lungs), which will be then used for classification (Jadwaa, 2022). Other techniques used are Thresholding and Noise reduction. Jadwaa used Otsu's method for global thresholding, which gives us

the binarized image. Then the image was filtered using a Median filter to reduce the noise of thresholding. Then Canny Edge detector follows.

In the article Gilewski (2022), introduced Hough transform, a technique used to identify straight lines in an image (Gilewski, 2022). It is usually partnered up with an edge detector algorithm (e.g. Canny Edge). Additionally, Gilewski discussed morphological operations, such as dilation, erosion, opening, and closing. Next for noise reduction, he used a Sobel filter.



Implemented Computer Vision Techniques

In this activity, we used a combination of techniques from the two stated literature, to accomplish the main objective. The first technique used is Otsu's thresholding so that the image can be easily processed and the ROI is easy to detect. After that, we used a Median filter to reduce noise generated by Otsu's thresholding. The result of the median filter is used as input for Canny edge detection. Then the edges detected were input to Hough transformation to identify the ROI boundaries in the form of small lines. After that, dilation, a morphological operation, is used to brighten up the ROI boundary. Then to visualize the ROI boundary, a contour was used.

Finally, we will be using the Jaccard Similarity Index to determine the closeness of the resulting image to the original image. Then the original image, detected edge, and the result of Hough Transformation will be displayed.

Solution Implementation

Before coding the solution these are the requirements needed to implement the solution.

1. Access to Google Collaboratory
2. Install Library Dependencies
    a. Pydicom
    b. Numpy
    c. Skimage
    d. Matplotlib
    e. Open CV Python

The steps identified in the “Implemented Computer Vision Techniques” Diagram are implemented by creating functions.

Technique	Description	Usage	Parameters	Return
Otsu’s Threshold	Converts grayscale image to binary form (Jadwaa, 2022)	apply_otsu_threshold(image)	image (numpy.ndarray)	Binary image - (numpy.ndarray)
Median Filter	Remove noise from an image (Jadwaa, 2022)	apply_median_filter(image, kernel_size)	• image (numpy.ndarray) • kernel_size (int, def = 50)	Filtered image (numpy.ndarray)
Canny edge detector	It effectively combines Gaussian smoothing, gradient computation, nonmaxima suppression, and double-threshold detection for accurate edge detection. (Jadwaa,2022)	apply_canny_edge_detector(image, low_threshold, high_threshold)	• image (numpy.ndarray) • low_threshold (int, def = 1) • high_threshold (int, def = 1)	Edges (numpy.ndarray)
Hough transformation and contour	Tries to detect straight lines in an image. Usually take input of the output of Canny edge detector (Gilewski, 2022)	hough_line(median_filtered_image, image_copy)	• median_filtered_image (numpy.ndarray) – result image after median filter • image_copy (numpy.ndarray)	Line Image (numpy.ndarray) – original image merge with the result of Hough transformation
Dilation	A morphological process that make lines thicker (Gilewski, 2022). It was used with Hough transformation and contour	post_process(edge)	edge (numpy.ndarray) – result image of canny edge detector	Result (numpy.ndarray)
Evaluation	Evaluates the closeness of images using Jaccard similarity index	eval(image1, image2)	• image1 (numpy.ndarray) • image1 (numpy.ndarray)	Jaccard Similarity Index (float)

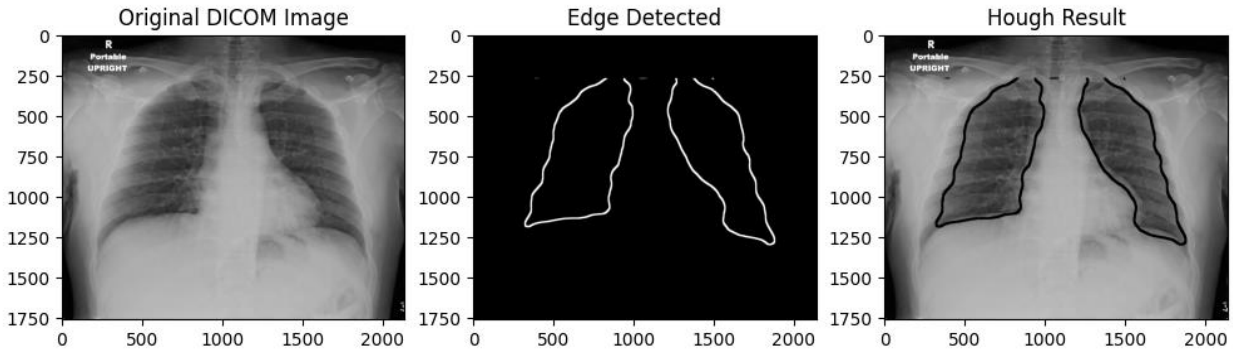
Sample Case

Image used (9a10650e-5e39-474b-a741-72b6ae0a311f.dcm) was downloaded from TCIA. It is an X-ray Chest PA of a patient.

```
def main():
    #load image
    dicom_path = '/content/9a10650e-5e39-474b-a741-72b6ae0a311f.dcm'
    dicom_image = load_dicom_image(dicom_path)

    #apply threshold
    thresholded_image = apply_otsu_threshold(dicom_image)
    median_filtered_image = apply_median_filter(thresholded_image)

    #get edges using canny, use hough transformation, add morphology (dilation) and contour
    hough = hough_line(median_filtered_image,dicom_image)

    #evaluate result using jaccard index
    score = eval(dicom_image,hough)
    print("Jaccard INdex:",score)

    #display orig, edge detected, and hough result
    temp = apply_canny_edge_detector(median_filtered_image)
    temp = post_process(temp)
    images = [dicom_image, temp,hough]
    titles = ['Original DICOM Image','Edge Detected','Hough Result']

    plot_images(images, titles)
```

Testing out the function with actual image

The image below shows Jaccard Similarity Index of original and result image, and three images. The three images are original input image, the edge detcteed by Canny edge detector, and the result of Hough transformation.



References

1. Jadwaa, S. A. K. (2022). X-Ray Lung Image Classification Using a Canny Edge Detector. Journal of Electrical and Computer Engineering, 2022

2. Gilewski, S. (2022, May 2). Edge detection and processing using Canny edge detector and Hough transform. *Wunderman Thompson*. https://wttech.blog/blog/2022/edge-detection-and-processing-using-canny-edge-detector-and-hough-transform/